UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/872,435	06/01/2001	Stephen L. Bade	SYN-0472	2097

35273 7590 07/13/2007

BEVER, HOFFMAN & HARMS, LLP
2099 GATEWAY PLACE
SUITE 320
SAN JOSE, CA 95110

EXAMINER	
LUU, CUONG V	
ART UNIT	**PAPER NUMBER**
2128	

MAIL DATE	DELIVERY MODE
07/13/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 09/872,435
Filing Date: June 01, 2001
Appellant(s): BADE ET AL.

Jeanette Harms
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 3/12/2007 appealing from the Office action mailed 12/18/2006.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings, which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

- Rompaey (E.U. Application 96870126.8)
- Cadence (Datasheet Interactive Simulation Library, 072597CM5.97, 1997)
- Hellestrand et al (U.S. Patent 6263302 B1)
- Schwab (U.S. Pub. 2004/0250083 A1)
- Van Huben et al (U.S. Patent 6094654)
- Schubert et al (U.S. Pub. 2005/0193280 A1)

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claims 37-39 and 49-51 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rompaey (E.U. Application 96870126.8) in view of Cadence (Datasheet Interactive Simulation Library, 072597CM5.97, 1997), and further in view of Hellestrand et al (U.S. Patent 6263302 B1).

1. As per claim 37, Rompaey teaches in a computer system having a graphical interface and a design language for forming a finite state machine (FSM) representation of a hardware partition of an embedded system, a method of designing an embedded system, the method comprising:

   forming a library of processors including an instruction set accurate simulator for each of the processor cores in the library (col. 9, lines 19-27);

responsive to a first sequence of user commands, selecting at least one of the processor cores from the library as a target processor core (col. 9, lines 23-25);

responsive to a second sequence of user commands, forming a virtual embedded system including an instruction set accurate simulator of a target processor core coupling read, write, and interrupt signals of the instruction set accurate simulator with an FSM simulation of at least one hardware element, wherein generating said FSM ... defining the behavior of the graphical symbol (col. 9, lines 14-27; col. 20, lines 55-58; col. 21, lines 1-14; p. 8, col. 11, lines 5-6; p. 31, Fig. 11. The recited lines 5-6 on page 8 indicate schematic of a design shown in Fig. 11. The schematic shows symbols of the design with user-definable texture portions defining the behavior of the graphical symbols. For example, in Fig. 11, the graphical symbol 104 includes texts 102, 103, and 105 defining its behavior);

responsive to a request from the user, loading an executable binary file of a software application compiled for the target processor core (col. 13, lines 44-52);

executing a simulation of the virtual embedded system running the software application (col. 21, lines 43-47);

Rompaey does not teach:

responsive to a user request, displaying on the GUI a graphical representation of the execution of the software application on the virtual embedded system that includes a software debugger interface to debug the loaded software and a virtual test-bench associated with the GUI and adapted to interact with the simulation, wherein the virtual test-bench is created using a test-bench builder for generating a graphical representation of at least one interactive test-bench and for selecting signals or variables associated with the FSM and to be coupled to a graphical representation of a user interface for each interactive test bench, to emulate user input to and device output from the virtual embedded system.

Hellestrand et al teach responsive to a user request, displaying on the GUI a graphical representation of the execution of the software application on the virtual embedded system that includes a software debugger interface to debug the loaded software (col. 21, lines 39-59).

Cadence teaches a virtual test-bench associated with the GUI and adapted to interact with the simulation, wherein the virtual test-bench is created using a test-bench builder for generating a graphical representation of at least one interactive test-bench and for selecting signals or variables associated with the FSM and to be coupled to a graphical representation of a user interface for each interactive test bench, to emulate user input to and device output from the virtual embedded system (p. 2, paragraphs 1-3.)

It would have been obvious to one of ordinary skill in the art to combine the teachings of Rompaey, Cadence and Hellestrand et al. Cadence's and Hellestrand et al's teachings would have provided interactive control of the simulation and visual examination of outcome rather than in "batch-mode" and dramatically accelerated the integration, test and evaluation phase of product development (Cadence, p. 3, paragraph 1).

2. As per claim 38, Rompaey teaches a design language including a plurality of graphical symbols with each graphical symbol having a graphical semantic portion and a textual semantic portion (col. 13, lines 56-58; col. 14, lines 1-4).

3. As per claim 39, Rompaey teaches Verilog design language inheriting these features below (col. 12, lines 9-11):

a start object defining a starting point of the finite state machine at an initialization time, the start object having an output connector activated when the start object is initialized;

a state object for representing a state of the finite state machine;

a decision object having an evaluation field for directing a flow of execution based on a result of an expression in the decision field;

a signal-out object for sending a communication signal; a signal-in object for receiving a communication signal;

a connector object for connecting control flow;

a process object for representing a finite state machine process.

But not these features which Rompaey also teach:

a task object including a field for inputting computer code in the C language for defining a behavior of the task object and a connector port for coupling the task object to other objects (col. 13, lines 56-58; col. 14, lines 1-4);

a symbol object having at least one user-definable pin connector and containing a block or process object: a block object for describing the behavior of one or more processes (col. 13, lines 56-58; col. 14, lines 1-4).

4. As per claim 49, Hellestrand et al teach the graphical user interface comprising:

responsive to a user input, associating a breakpoint of execution with a graphical symbol (col. 21, lines 39-40 and 57-59);

receiving a request to debug software (col. 21, lines 28-37); and

stopping the simulation responsive to a command flow of the FSM representation of the hardware element reaching the graphical symbol of the breakpoint of execution (col. 21, lines 42-44 and 57-59).

5. As per claim 50, Hellestrand et al teach responsive to a user request, single-stepping the simulation to sequential breakpoints of execution in the command flow of the FSM representation of hardware elements (col. 21, lines 53-56).

6. As per claim 51, Hellestrand et al teach responsive to a user request, single-stepping the simulation by a pre-selected number of time units (col. 21, lines 53-56).

Claims 40-42 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rompaey (E.U. Application 96870126.8) in view of Cadence and Hellestrand et al as applied to claim 37 above, and further in view of Schwab (U.S. Pub. 2004/0250083 A1).

7. As per claim 40, Rompaey teaches storing the virtual embedded system as a design (col. 17, lines 49-54),

   but not in a design repository coupled to a server; and

   providing access privileges to the design to a selected individual or group.

   Cadence and Hellestrand et al do not teach this feature either.

   Schwab teaches these features (the abstract).

   It would have been obvious to one of ordinary skill in the art to combine the teachings of Rompaey, Cadence, Hellestrand et al, and Schwab. Schwab's teachings would have provided access to a requester, possibly not physically located at the remote terminal, only after verifying that the requester is authorized to view the items (p. 1, paragraph 0002).

8. As per claim 41, Rompaey, Cadence and Hellestrand et al do not teach responsive to a user command, providing access privileges to the design to a group of vendors.

Schwab teaches this feature (the abstract).

It would have been obvious to one of ordinary skill in the art to combine the teachings of Rompaey, Cadence, Hellestrand et al, and Schwab. Schwab's teachings would have provided access to a requester, possibly not physically located at the remote terminal, only after verifying that the requester is authorized to view the items (p. 1, paragraph 0002).

9. As per claim 42, Rompaey, Cadence and Hellestrand et al do not teach the design is accessible from an on-line bidding board.

Schwab teaches products accessible from an on-line bidding board (p. 10, paragraph 0093).

It would have been obvious to one of ordinary skill in the art to combine the teachings of Rompaey, Hellestrand et al, Van Huben et al, and Schwab. Schwab's teachings would have made the design available to potential buyers online.

Claims 43-48 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rompaey (E.U. Application 96870126.8) in view of Cadence and Hellestrand et al as applied to claim 37 above, and Van Huben et al (U.S. Patent 6094654).

10. As per claim 43, this limitation has been discussed in claim 41. It is, therefore, rejected for the same reason.

11. As per claim 44, this limitation has been discussed in claim 41. It is, therefore, rejected for the same reason.

12. As per claim 45, Rompaey and Hellestrand et al do not teach a design manager application permitting one or more members of a design team to select edit privileges of at least one version of the design.

Van Huben et al teach this feature (col. 16, lines 19-22; col. 29, lines 54-59).

It would have been obvious to one of ordinary skill in the art to combine the teachings of Rompaey, and Hellestrand et al, and Van Huben et al. Van Huben et al's teachings would have provided selected users capability to access a specific version of a design.

13. As per claim 46, Rompaey and Hellestrand et al do not teach permitting one member of the design team to load and execute a binary program executable of a software application compiled for the target processor core onto the design.

Van Huben et al teach this feature (col. 15, lines 48-55).

It would have been obvious to one of ordinary skill in the art to combine the teachings of Rompaey, and Hellestrand et al, and Van Huben et al. Van Huben et al's teachings would have enabled designers to promote data or run library processes without the need for a data manager to process it.

14. As per claim 47, Rompaey and Hellestrand et al do not teach providing version control and regulating editing access to maintain a consistent version of the design.

Van Huben et al teach these features (col. 16, lines 19-22).

It would have been obvious to one of ordinary skill in the art to combine the teachings of Rompaey, and Hellestrand et al, and Van Huben et al. Van Huben et al's teachings would have provided selected users capability to access a specific version of a design.

15. As per claim 48, these limitations have already been discussed in claim 40. They are, therefore, rejected for the same reasons.

Claims 52-59, 64, and 89 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rompaey in view of Cadence.

16. As per claim 52, Rompaey teaches a computer implemented method of embedded system design, the method comprising:

selecting an instruction set accurate simulator of a target processor core (col. 9, lines 19-27 and 23-25);

generating a virtual hardware component that is a finite state machine (FSM) representation of at least one hardware component ... defining the behavior of the graphical symbol (col. 9, lines 14-27; col. 20, lines 55-58; col. 21, lines 1-14; p. 8, col. 11, lines 5-6; p. 31, Fig. 11. The recited lines 5-6 on page 8 indicate schematic of a design shown in Fig. 11. The schematic shows symbols of the design with user-definable texture portions defining the behavior of the graphical symbols. For example, in Fig. 11, the graphical symbol 104 includes texts 102, 103, and 105 defining its behavior);

linking read, write, and interrupt signals of the instruction set accurate simulator of the target processor core with corresponding signals of the at least one virtual hardware component to form a virtual embedded system (col. 9, lines 19-22; col. 20, lines 55-58; col. 21, lines 1-14);

coupling a virtual test bench to at least one signal or variable of the virtual embedded system to simulate a human/machine interface (col. 21, lines 8-16); and

coupling a software debugger to the virtual embedded system that is configured to load and run on the virtual embedded system at least one binary program executable of a software application compiled for the target processor core (col. 13, lines 39-48. The applicant mentions Coware, and it inherits software debugger feature).

But does not teach:

Creating a virtual test bench ... to be coupled to a graphical representation of a user interface for each interactive test bench.

Cadence teaches this limitation (p. 2, paragraphs 1-3).

It would have been obvious to one of ordinary skill in the art to combine the teachings of Rompaey and Cadence. Cadence's teachings would have provided interactive control of the simulation and visual examination of outcome rather than in "batch-mode" and dramatically accelerated the integration, test and evaluation phase of product development (Cadence, p. 3, paragraph 1).

17. As per claim 53, Rompaey teaches selecting the target processor core from a library having a plurality of instruction set accurate simulators for a plurality of processor cores (col. 9, lines 19-27).

18. As per claim 54, Rompaey teaches selecting the virtual hardware component from a library of virtual hardware components (col. 9, lines 23-25).

19. As per claim 55, Rompaey teaches modifying the virtual hardware component (col. 7, lines 43-55).

20. As per claim 56, Rompaey teaches loading benchmark software in an evaluation phase of an embedded system project and running a simulation of the virtual embedded system executing the benchmark software (col. 10, lines 10-13. Benchmark software is a type of application software. Therefore, the examiner interprets Rompaey suggests a benchmark software to be executed on the virtual embedded system).

21. As per claim 57, Rompaey teaches loading binary program executables of development software compiled for the target processor core in a development phase of an embedded systems project and running a simulation of the virtual embedded system executing the development software (col. 10, lines 10-13).

22. As per claim 58, this limitation has already been discussed in claim 52. It is, therefore, rejected for the same reasons.

23. As per claim 59, Rompaey teaches storing the virtual embedded system as a design having at least one executable file in a design repository (col. 17, lines 49-54l col. 32, lines 41-47).

24. As per claim 64, Rompaey teaches a method of designing an embedded system, the method comprising:

defining a system architecture of the embedded system (col. 7, lines 31-35);

generating a finite state machine (FSM) representation of at least one hardware element ... defining the behavior of the graphical symbol (col. 9, lines 14-27; col. 20, lines 55-58; col. 21, lines 1-14; p. 8, col. 11, lines 5-6; p. 31, Fig. 11. The recited lines 5-6 on page 8 indicate

schematic of a design shown in Fig. 11. The schematic shows symbols of the design with user-definable texture portions defining the behavior of the graphical symbols);

designing a virtual prototype of the embedded system having an instruction set accurate simulator of a target processor core coupling read, write, and interrupt signals of the instruction set accurate simulator with said FSM representation of at least one hardware element (col. 20, lines 55-58; col. 21, lines 1-14);

coupling the virtual prototype to a software debugger having a debugging interface and to the virtual test bench (col. 9, lines 19-22; col. 13, lines 39-48. The applicant mentions Coware, and it inherits graphical interface with the embedded system);

developing at least one software application for the processor core (col. 10, lines 10-13);

loading compiled binary program code compiled from the at least one software application for execution of the virtual prototype (col. 10, lines 10-13); and

initiating a simulation of the virtual prototype executing the at least one software application (col. 10, lines 10-13).

But does not teach:

Creating a virtual test bench ... to be coupled to a graphical representation of a user interface for each interactive test bench.

Cadence teaches this limitation (p. 2, paragraphs 1-3).

It would have been obvious to one of ordinary skill in the art to combine the teachings of Rompaey and Cadence. Cadence's teachings would have provided interactive control of the simulation and visual examination of outcome rather than in "batch-mode" and dramatically accelerated the integration, test and evaluation phase of product development (Cadence, p. 3, paragraph 1).

25. As per claim 89, Cadence teaches the step of interacting at run time with the virtual prototype to simulate an application of the embedded system (p. 2, paragraph 3. In this paragraph Cadence teaches the interactive control when running the simulation. This is regarded as interacting at run time with the virtual prototype to simulate an application of the embedded system).

Claims 60-63 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rompaey as applied to claims 52 and 59 above, and further in view of Van Huben et al.

26. As per claim 60, these limitations have already discussed in claim 40. They are, therefore, rejected for the same reasons.

27. As per claim 61, Rompaey does not teach user-group being a geographically distributed embedded system project team.

Huben et al teach this feature (col. 12, lines 61-67; col. 13, lines 10-14).

It would have been obvious to one of ordinary skill in the art to combine the teachings of Rompaey, and Van Huben et al. Van Huben et al's teachings would have enforced provision of a computing environment that can handle distributed computing.

28. As per claim 62, Rompaey does not teach providing a version of the design to a vendor offering a service related to the virtual embedded system.

Huben et al teach this feature (col. 15, lines 48-55).

It would have been obvious to one of ordinary skill in the art to combine the teachings of Rompaey, and Van Huben et al. Van Huben et al's teachings would have allowed vendors to run library processes without the need for a Data Manager to process it.

29. As per claim 63, Rompaey does not teach the design being stored on a server and a vendor being provided access to the design via network connection.

Huben et al teach this feature (col. 15, lines 48-55).

It would have been obvious to one of ordinary skill in the art to combine the teachings of Rompaey, and Van Huben et al. Van Huben et al's teachings would have provided access to vendors at distant locations.

Claim 65-67 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rompaey in view of Schubert et al (U.S. Pub. 2005/0193280 A1).

30. As per claim 65, Rompaey does not teach developing a hardware implementation using the virtual prototype as a functional specification describing a hardware partition.

Schubert et al teach this feature (p. 2, paragraph 0017).

It would have been obvious to one of ordinary skill in the art to combine the teachings of Rompaey and Schubert et al. Schubert et al's teachings would have enabled running testbench software faster than on virtual prototype.

31. As per claim 66, Rompaey teaches:

evaluating the operation of the embedded system executing the at least one software application (col. 10, lines 10-13); and

responsive to a result of the evaluation, modifying a hardware component of the embedded system (col. 7, lines 43-55).

32. As per claim 67, Rompaey teaches:

selecting at least one embedded system component for evaluation (col. 10, lines 10-13);

forming a virtual evaluation platform including the selected embedded system component (col. 10, lines 10-13);

loading a benchmark software application for execution on the virtual evaluation platform (col. 10, lines 10-13. Benchmark software is a type of application software. Therefore, the examiner interprets Rompaey suggests a benchmark software to be executed on the virtual embedded system); and

evaluating performance of the virtual evaluation platform executing the benchmark software application (col. 10, lines 10-13).

Claims 68-70 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rompaey in view of Cadence and Van Huben et al.

33. As per claim 68, Rompaey teaches a method of providing information to potential suppliers for procuring a good or service associated with an embedded system, the method comprising:

defining a system architecture of the embedded system (col. 7, lines 31-35);

designing a virtual prototype of the embedded system, the virtual prototype having an instruction set accurate simulator of a target processor core, wherein generating said FSM

... a user-definable texture portion defining the behavior of the graphical symbol (col. 20, lines 55-58; col. 21, lines 1-14; p. 8, col. 11, lines 5-6; p. 31, Fig. 11);

coupling the virtual prototype to a virtual test bench having a graphical representation of a human/machine interface for interacting with the embedded system (col. 9, lines 19-22; col. 13, lines 39-48. The applicant mentions Coware, and it inherits graphical interface with the embedded system);

Rompaey does not teach:

creating a virtual test bench ... a user interface for each interactive test bench; and

publishing the virtual prototype as a functional specification from which a vendor may initiate a simulation of the operation of the embedded system.

Cadence teaches creating a virtual test bench ... a user interface for each interactive test bench (p. 2, paragraphs 1-3);

Van Huben et al teach making a design available to a selected individual or group to dispatch tasks on it (col. 29, lines 54-59; col. 15, lines 48-55).

It would have been obvious to one of ordinary skill in the art to combine the teachings of Rompaey, Cadence and Van Huben et al. Cadence's and Van Huben et al's teachings would have provided interactive control of the simulation and visual examination of outcome rather than in "batch-mode" and dramatically accelerated the integration, test and evaluation phase of product development and access of the stored embedded system for reuse to a selected group or people.

34. As per claim 69, storing a design on a computer readable medium and publishing it by sending the computer readable medium to a vendor is very well known. These limitations are, therefore, rejected.

35. As per claim 70, Rompaey does not teach the virtual prototype being published by posting the virtual prototype as a design hosted on a database coupled to the network server.

Van Huben et al teach this feature (col. 12, 46-67; col. 13, lines 1-12).

It would have been obvious to one of ordinary skill in the art to combine the teachings of Rompaey, and Van Huben et al. Van Huben et al's teachings would have provided access of the stored embedded system to users at distant locations.

Claims 71-72 are rejected under 35 U.S.C. 103(a) as being unpatentable over Rompaey in view of Cadence and Van Huben et al as applied to claims 68 and 70 above, and further in view of Schwab.

36. As per claim 71, Rompaey and Van Huben et al do not teach the virtual prototype is published to a bulletin board of the database.

Schwab teaches this feature (p. 10, paragraph 0093).

It would have been obvious to one of ordinary skill in the art to combine the teachings of Rompaey, Van Huben et al, and Schwab. Schwab's teachings would have made the design available to potential buyers online.

37. As per claim 72, Rompaey and Van Huben et al do not teach the bulletin board being a bidding board.

Schwab teaches this feature (p. 10, paragraph 0093).

It would have been obvious to one of ordinary skill in the art to combine the teachings of Rompaey, Van Huben et al, and Schwab. Schwab's teachings would have made the design available to potential buyers to bid online.

Claims 74, 76-78 are rejected under 35 U.S.C. 103(a) as being unpatentable over Van Huben et al in view of Cadence and Rompaey.

38. As per claim 74, Van Huben et al teach a computer-implemented method for a vendor to acquire information for the procurement of a service related to a project, the method comprising:

accessing a database of a design (col. 29, lines 55-59);

instantiating an instance of one of the virtual prototypes (col. 15, lines 48-53); and

evaluating the virtual prototype (col. 15, lines 48-53).

But Van Huben et al do not teach:

The design being a virtual prototype of embedded system, each of the virtual prototypes having a processor simulator, a finite state machine representation of hardware peripherals, and a virtual test bench emulating a human/machine interface for interacting with a simulation of the operation of the virtual prototype.

Wherein generating said FSM representation ... defining the behavior of the graphical symbol, and

Wherein the virtual test bench ... a user interface for the interactive test bench;

Rompaey teaches the design being a virtual prototype of embedded system, each of the virtual prototypes having a processor simulator, a finite state machine representation of hardware peripherals, and a virtual test bench emulating a human/machine interface for

interacting with a simulation of the operation of the virtual prototype (col. 20, lines 55-58; col. 21, lines 1-14; col. 9, lines 19-22; col. 13, lines 39-48).

Wherein generating said FSM representation ... defining the behavior of the graphical symbol (p. 8, col. 11, lines 5-6; p. 31, Fig. 11).

Cadence teaches wherein the virtual test bench ... a user interface for the interactive test bench (p. 2, paragraphs 1-3).

It would have been obvious to one of ordinary skill in the art to combine the teachings of Van Huben et al, Rompaey and Cadence. Rompaey's and Cadence's teachings would have made a virtual prototype of embedded system accessible for evaluation and provided interactive control of the simulation and visual examination of outcome rather than in "batch-mode" and dramatically accelerated the integration, test and evaluation phase of product development.

39. As per claim 76, Van Huben et al teach the design configured to show a parts list (col. 20, lines 5-8) but do not teach the design being the virtual prototype.

Rompaey teaches this feature (col. 20, lines 55-58; col. 21, lines 1-14; col. 9, lines 19-22; col. 13, lines 39-48).

It would have been obvious to one of ordinary skill in the art to combine the teachings of Van Huben et al and Rompaey. Rompaey's teachings would have made the part list available to users to track their synchronization.

40. As per claim 77, Van Huben et al teach the design configured to show a component net list (col. 151, lines 6-9) but do not teach the design being the virtual prototype.

Rompaey teaches this feature (col. 20, lines 55-58; col. 21, lines 1-14; col. 9, lines 19-22; col. 13, lines 39-48).

It would have been obvious to one of ordinary skill in the art to combine the teachings of Van Huben et al and Rompaey. Rompaey's teachings would have allowed building models from released data.

41. As per claim 78, Van Huben et al teach the database of designs being hosted on a network server accessible by a client computer (col. 12, 46-67; col. 13, lines 1-12).

Claims 75 is rejected under 35 U.S.C. 103(a) as being unpatentable over Van Huben et al in view of Cadence and Rompaey as applied to claim 74 above, and further in view of Schwab.

42. As per claim 75, Van Huben et al and Rompaey do not teach submitting a quote for a good or service related to the embedded system simulated by the virtual prototype.

Schwab teaches submitting a quote for a good or service related to a product (p. 4, paragraph 0038).

It would have been obvious to one of ordinary skill in the art to combine the teachings of Rompaey, Van Huben et al, and Schwab. Schwab's teachings would have allowed bid on a good or service related to the embedded system simulated by the virtual prototype.

(10) Response to Argument

1. The Appellant's arguments, regarding claim 37, see pages 18-21 of Appeal Brief, have been fully considered but they are not persuasive. The Appellant argues that Rompaey does not

teach "forming the virtual embedded system including an instruction set accurate simulator of a target processor core and coupling read, write, and interrupt signals of the instruction set accurate simulator with an FSM simulation of at least one hardware element". The Examiner respectfully disagrees. At col. 9 lines 14-27, Rompaey recites:

> Said simulation can be a multi-platform simulation being executed on a plurality of computers.
> Said simulation can be a hybrid simulation comprising substantially simultaneous hardware implementations and computer simulations.
> Said implementation can be heterogeneous implementation comprising hardware subsystems and software subsystems, said software subsystems being executed on one on more of said hardware subsystems.
> Said hardware subsystems can comprise any one or more of processor cores, off-the-shelf components, custom components, ASICs, processors, or boards.
> Said software subsystems can comprise machine instructions for said hardware subsystems

and at col. 20 lines 55-58 and col. 21 lines 1-14:

> All ports (75) have primitive protocols. The software model identifies, for example, what ports can be used to get data in or out of the processor core (memory mapped, co-processor port ...) what ports can be used as interrupt ports and what their characteristics are (interrupt priority, maskable interrupt ...). In addition the software model contains a behavioral description that allows to compile a software host language encapsulation into machine code. For example: functions to manage processor specific actions such as installing an interrupt vector, enabling/disabling interrupts, etc. In Figure 7, the software model of the ARM-6 RISC processor is shown. A number of its ports (75) are shown in Figure 7. The memory port mem is modeled as a (bi-directional) slave port. This slave port is accessed by the device drivers, by means of an RPC, to write/read data to/from the external hardware.

Rompaey teaches, as disclosed on col. 9 lines 14-27, hardware subsystems comprising processor cores for simulation. A processor core is made up of the control unit and arithmetic logic unit. The control unit is further divided into 2 sub units, one of which is a finite state machine (FSM), which is used to control sequences of all actions performed and elements for producing proper control signals for difference elements. Rompaey also teaches machine instructions for said hardware subsystems. This teaching in combination with the teaching of processor cores for hardware subsystems read onto the limitation of an instruction set accurate simulator of a target processor core. Col. 20 lines 55-58 and col. 21

lines 1-14 indicate that Rompaey teaches coupling read, write, and interrupt signals of the instruction set accurate simulator with an FSM simulation of at least one hardware element.

The Appellant further argues that Rompaey does not teach "wherein generating said FSM simulation comprises applying a design language having at least one graphical symbol and adapted to form a finite state machine representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol."

Rompaey recites Fig. 11 on page 31 (attached at the end of this Response to Argument section), text on col. 11, lines 5-6:

> Figure 11 is a schematic representation of the pager application as described with the present invention.

Figure 11 in combination with recitation at col. 9 lines 14-27 clearly teaches graphical symbol and adapted to form a finite state machine representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion inside the symbol defining the behavior of the graphical symbol. For example, graphical symbol 97 (bottom right corner) represents Phase Correction block of a design with design language textual portion 110 inside it to define its behavior.

The Appellant also argues that neither Hellestrand nor Cadence teaches the recited step of displaying on the GUI a graphical representation of the execution of the software application on the virtual embedded system that includes a software debugger interface to debug the loaded software and. The Examiner respectfully disagrees.

Hellestrand recites on col. 21 lines 39-59:

> Prior to execution, the user may insert debugger breakpoints in the user programs for each processor simulator. Prior to execution the user can enable or disable the breakpoints. As the simulation is run under debugger control, whenever a breakpoint is encountered, the debugger stops

execution. At this point, any software variable in any of the processor simulators and any hardware variable in the hardware simulator may be examined. The particular implementation of the invention provides a window on the viewer screen for each of the processor simulators and for the hardware simulator. When the systems stops at a breakpoint, the current instruction is highlighted. In addition, the implementation provides a "voltmeter"-like "value watch" window, at user option, to examine any hardware entities as the simulation proceeds. The environment also provides for single stepping both the processor simulators one instruction at a time, and the hardware simulator any number of time units at a time.

The preferred embodiment environments provides, at user option, both a command line operating mode, and an interactive mode under a graphical user interface.

The recited paragraphs above by Hellestrand clearly indicate a software debugger interface to debug the loaded software with the GUI and adapted to interact with the simulation.

The Appellant also argues that Cadence does not teach a virtual test-bench associated with the GUI and adapted to interact with the simulation, wherein the virtual test-bench is created using a test-bench builder for generating a graphical representation of at least one interactive test-bench and for selecting signals or variables associated with the FSM and to be coupled to a graphical representation of a user interface for each interactive test-bench, to emulate user input to and device output from the virtual embedded system.

Cadence recites on p. 2, paragraphs 1-3:

The Interactive Simulation Library makes constructing VIs of any kind simple and straightforward. ISL control windows can display scroll bars, push-buttons, pull-down menus and more for changing gains, frequencies, noise-levels and other design parameters. ISL display windows show real-time oscillograms, bar graphs, eye diagrams, and waterfall plots indicating the effects the controls have on the system.

You specify features and displays for an interactive control window by choosing ISL primitives (basic function blocks) from the library and wiring them together on the workstation screen (hierarchy is allowed). The resulting block diagram is a complete blueprint for the window; no written programming is required. These capabilities enable you to define whatever kind of signal post-processing is required to display the information you want.

Once the ISL block diagram is specified, you incorporate it into the signal flow block diagram for the system under test. When you run a simulation, the interactive control and display windows you created pop up automatically and begin monitoring the systems performance.

In the recited paragraphs above, Cadence teaches a test-bench, a tool to test a design, that a user can interactively select signals on a design, which can be a FSM when combined with Rompaey's teachings, to be post-processed and displayed after running a simulation.

It would have been obvious to one of ordinary skill in the art to combine the teachings of Rompaey, Cadence and Hellestrand et al. Cadence's and Hellestrand et al's teachings would have provided, at user option, both a command line operating mode, and an interactive mode under a graphical user interface (Hellestrand, col .21 lines 57-59) and interactive control of the simulation and visual examination of outcome rather than in "batch-mode" and dramatically accelerated the integration, test and evaluation phase of product development (Cadence, p. 3, paragraph 1).

Claim 37, therefore, is rejected.

2. As per claims 38-39 and 49-51, the Appellant argues that they are allowed since they depend on allowed claim 37, see p. 21. Since claim 37 remains rejected, claims 38-39 and 49-51 remain rejected.

3. As per claims 40-42, the Appellant argues that they are allowed since they depend on allowed claim 37, see p. 22. Since claim 37 remains rejected, claims 40-42 remain rejected.

4. As per claims 43-48, the Appellant argues that they are allowed since they depend on allowed claim 37, see pp. 22-24. Since claim 37 remains rejected, claims 43-48 remain rejected.

5. The Appellant's arguments, regarding claim 52, see pages 24-27 of Appeal Brief, have been fully considered but they are not persuasive. The Appellant argues that Rompaey does not teach:

generating a virtual hardware component that is a finite state machine (FSM) representation of at least one hardware component,

linking read, write, and interrupt signals of the instruction set accurate simulator of the target processor core with corresponding signals of the at least one virtual hardware component to form a virtual embedded system.

The Examiner respectfully disagrees. At col. 9 lines 14-27, Rompaey recites:

> Said simulation can be a multi-platform simulation being executed on a plurality of computers.
> Said simulation can be a hybrid simulation comprising substantially simultaneous hardware implementations and computer simulations.
> Said implementation can be heterogeneous implementation comprising hardware subsystems and software subsystems, said software subsystems being executed on one on more of said hardware subsystems.
> Said hardware subsystems can comprise any one or more of processor cores, off-the-shelf components, custom components, ASICs, processors, or boards.
> Said software subsystems can comprise machine instructions for said hardware subsystems

and at col. 20, lines 55-58 and col. 21, lines 1-24:

> All ports (75) have primitive protocols. The software model identifies, for example, what ports can be used to get data in or out of the processor core (memory mapped, co-processor port ...) what ports can be used as interrupt ports and what their characteristics are (interrupt priority, maskable interrupt ...). In addition the software model contains a behavioral description that allows to compile a software host language encapsulation into machine code. For example: functions to manage processor specific actions such as installing an interrupt vector, enabling/disabling interrupts, etc. In Figure 7, the software model of the ARM-6 RISC processor is shown. A number of its ports (75) are shown in Figure 7. The memory port mem is modeled as a (bi-directional) slave port. This slave port is accessed by the device drivers, by means of an RPC, to write/read data to/from the external hardware. The slave thread, modeled in the software model, attached to the mem slave port translates the incoming RPC to a memory access. SYMPHONY makes the connection between the device drivers and the mem port. The fiq port is modeled as a master port. The software model of the ARM processor ensures that an RPC to this port is performed, every time the processor detects that an interrupt has occurred. SYMPHONY connects the fiq port to a slave thread that serves as the interrupt service routine, so that routine is started automatically.

Rompaey teaches, as disclosed on col. 9 lines 14-27, hardware subsystems comprising processor cores for simulation. A processor core is made up of the control unit and arithmetic logic unit. The control unit is further divided into 2 sub units, one of which is a finite state machine (FSM), which is used to control sequences of all actions performed and elements for producing proper control signals for difference elements. Rompaey also teaches machine instructions for said hardware subsystems. This teaching in combination with the teaching of processor cores for hardware subsystems read onto the limitation of an instruction set accurate simulator of a target processor core. Furthermore, col. 20 lines 55-58 and col. 21 lines 1-24 indicate that Rompaey teaches linking read, write, and interrupt signals of the instruction set accurate simulator with an FSM simulation of at least one hardware element and actually performing the functions of accessing, reading or writing, memory and interrupt in simulations to form a virtual embedded system.

The Appellant further argues that Rompaey does not teach "said generating comprising applying a design language having at least one graphical symbol and adapted to form an FSM representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol".

Rompaey recites Fig. 11 on page 31, text on col. 11, lines 5-6:

> Figure 11 is a schematic representation of the pager application as described with the present invention.

Figure 11 in combination with recitation at col. 9 lines 14-27 clearly teaches graphical symbol and adapted to form a finite state machine representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion inside the symbol defining the behavior of the graphical symbol. For

example, graphical symbol 97 (bottom right corner) represents Phase Correction block in a design with design language textual portion 110 inside it to define its behavior.

It would have been obvious to one of ordinary skill in the art to combine the teachings of Rompaey and Cadence. Cadence's teachings would have provided interactive control of the simulation and visual examination of outcome rather than in "batch-mode" and dramatically accelerated the integration, test and evaluation phase of product development (Cadence, p. 3, paragraph 1).

Claim 52, therefore, is rejected.

6. As per claims 53-59, the Appellant argues that they are allowed since they depend on allowed claim 52, see p. 27. Since claim 52 remains rejected, claims 53-59 remain rejected.

7. The Appellant's arguments, regarding claim 64, see pages 27-30 of Appeal Brief, have been fully considered but they are not persuasive. The Appellant argues that Rompaey does not teach "generating a finite state machine (FSM) representation of at least one hardware element, said generating comprising applying a design language having at least one graphical symbol and adapted to form a finite state machine representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol".

The Examiner respectfully disagrees. At col. 9 lines 14-27, Rompaey recites:

> Said simulation can be a multi-platform simulation being executed on a plurality of computers.
> Said simulation can be a hybrid simulation comprising substantially simultaneous hardware implementations and computer simulations.
> Said implementation can be heterogeneous implementation comprising hardware subsystems and software subsystems, said software subsystems being executed on one on more of said hardware subsystems.
> Said hardware subsystems can comprise any one or more of processor cores, off-the-shelf components, custom components, ASICs, processors, or boards.

Said software subsystems can comprise machine instructions for said hardware subsystems

and Fig. 11 on page 31, text on col. 11, lines 5-6:

Figure 11 is a schematic representation of the pager application as described with the present invention.

Rompaey, as disclosed on col. 9 lines 14-27, teaches hardware subsystems comprising processor cores for simulation. A processor core is made up of the control unit and arithmetic logic unit. The control unit is further divided into 2 sub units, one of which is a finite state machine (FSM), which is used to control sequences of all actions performed and elements for producing proper control signals for difference elements. Rompaey also teaches machine instructions for said hardware subsystems. This teaching in combination with the teaching of processor cores for hardware subsystems read onto the limitation of an instruction set accurate simulator of a target processor core.

Figure 11 in combination with recitation at col. 9 lines 14-27 clearly teaches graphical symbol and adapted to form a finite state machine representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion inside the symbol defining the behavior of the graphical symbol. For example, graphical symbol 97 (bottom right corner) represents Phase Correction design with design language textual portion 110 inside it to define its behavior.

The Appellant further argues that Rompaey does not teach "a virtual prototype of the embedded system having an instruction set accurate simulator of a processor core and coupling read, write, and interrupt signals of the instruction set accurate simulator with said FSM representation of at least one hardware element".

Rompaey recites at col. 20, lines 55-58 and col. 21, lines 1-14;

All ports (75) have primitive protocols. The software model identifies, for example, what ports can be used to get data in or out of the processor core (memory mapped, co-processor port ...) what ports can be used as interrupt ports and what their characteristics are (interrupt priority, maskable interrupt ...). In addition the software model contains a behavioral description that allows to compile a software host language encapsulation into machine code. For example: functions to manage processor specific actions such as installing an interrupt vector, enabling/disabling interrupts, etc. In Figure 7, the software model of the ARM-6 RISC processor is shown. A number of its ports (75) are shown in Figure 7. The memory port mem is modeled as a (bi-directional) slave port. This slave port is accessed by the device drivers, by means of an RPC, to write/read data to/from the external hardware.

Col. 20 lines 55-58 and col. 21 lines 1-14 indicate that Rompaey teaches linking read, write, and interrupt signals of the instruction set accurate simulator with an FSM simulation of at least one hardware element and actually performing the functions of accessing, reading or writing, memory and interrupt in simulations to form a virtual embedded system.

The Appellant also argues that Cadence does not teach creating a virtual test bench having a graphical representation of a human/machine interface for interacting with the embedded system using a test bench builder for generating a graphical representation of at least one interactive test bench and for selecting signals or variables associated with the FSM and to be coupled to a graphical representation of a user interface for each interactive test bench.

The Examiner respectfully disagrees. Cadence recites on p. 2, paragraphs 1-3:

The Interactive Simulation Library makes constructing VIs of any kind simple and straightforward. ISL control windows can display scroll bars, push-buttons, pull-down menus and more for changing gains, frequencies, noise-levels and other design parameters. ISL display windows show real-time oscillograms, bar graphs, eye diagrams, and waterfall plots indicating the effects the controls have on the system.

You specify features and displays for an interactive control window by choosing ISL primitives (basic function blocks) from the library and wiring them together on the workstation screen (hierarchy is allowed). The resulting block diagram is a complete blueprint for the window; no written programming is required. These capabilities enable you to define whatever kind of signal post-processing is required to display the information you want.

Once the ISL block diagram is specified, you incorporate it into the signal flow block diagram for the system under test. When you run a simulation, the interactive control and display windows you created pop up automatically and begin monitoring the systems performance.

In the above paragraphs Cadence teaches a test-bench, a tool to test a design, that a user can interactively select signals on a design, which can be a FSM when combined with Rompaey's teachings, to be post-processed and displayed after running a simulation.

It would have been obvious to one of ordinary skill in the art to combine the teachings of Rompaey and Cadence. Cadence's teachings would have provided an interactive control of the simulation and visual examination of outcome rather than in "batch-mode" and dramatically accelerated the integration, test and evaluation phase of product development (Cadence, p. 3, paragraph 1).

Claim 64, therefore, is rejected.

8. As per claim 89, the Appellant argues that they are allowed since it depends on allowed claim 64, see p. 30. Since claim 64 remains rejected, claims 89 remains rejected.

9. As per claims 60-63, the Appellant argues that they are allowed since it depends on allowed claim 52, see pp. 30-31. Since claim 52 remains rejected, claims 60-63 remain rejected.

10. As per claims 65-67, the Appellant argues that they are allowed since it depends on allowed claim 64, see p. 31. Since claim 64 remains rejected, claims 65-67 remain rejected.

11. The Appellant's arguments, regarding claim 68, see pages 31-35 of Appeal Brief, have been fully considered but they are not persuasive. The Appellant argues that Rompaey does not teach "designing a virtual prototype of the embedded system, the virtual prototype having an instruction set accurate simulator of a target processor core and a finite state machine (FSM) representation of a hardware element within the embedded system, the FSM

representation configured to couple memory read/write requests and interrupt signals with the instruction set accurate simulator of the target processor core".

The Examiner respectfully disagrees. At col. 9 lines 14-27, Rompaey recites:

> Said simulation can be a multi-platform simulation being executed on a plurality of computers.
> Said simulation can be a hybrid simulation comprising substantially simultaneous hardware implementations and computer simulations.
> Said implementation can be heterogeneous implementation comprising hardware subsystems and software subsystems, said software subsystems being executed on one on more of said hardware subsystems.
> Said hardware subsystems can comprise any one or more of processor cores, off-the-shelf components, custom components, ASICs, processors, or boards.
> Said software subsystems can comprise machine instructions for said hardware subsystems

and at col. 20, lines 55-58 and col. 21, lines 1-14;

> All ports (75) have primitive protocols. The software model identifies, for example, what ports can be used to get data in or out of the processor core (memory mapped, co-processor port ...) what ports can be used as interrupt ports and what their characteristics are (interrupt priority, maskable interrupt ...). In addition the software model contains a behavioral description that allows to compile a software host language encapsulation into machine code. For example: functions to manage processor specific actions such as installing an interrupt vector, enabling/disabling interrupts, etc. In Figure 7, the software model of the ARM-6 RISC processor is shown. A number of its ports (75) are shown in Figure 7. The memory port mem is modeled as a (bi-directional) slave port. This slave port is accessed by the device drivers, by means of an RPC, to write/read data to/from the external hardware.

Rompaey, as disclosed on col. 9 lines 14-27, teaches hardware subsystems comprising processor cores for simulation. A processor core is made up of the control unit and arithmetic logic unit. The control unit is further divided into 2 sub units, one of which is a finite state machine (FSM), which is used to control sequences of all actions performed and elements for producing proper control signals for difference elements. Rompaey also teaches machine instructions for said hardware subsystems. This teaching in combination with the teaching of processor cores for hardware subsystems read onto the limitation of an instruction set accurate simulator of a target processor core. Furthermore, the recited col. 20, lines 55-58 and col. 21, lines 1-14 above indicate that Rompaey teaches coupling read,

write, and interrupt signals of the instruction set accurate simulator with an FSM simulation of at least one hardware element and actually performing the functions of accessing, reading or writing, memory and interrupt in simulations. Rompaey does not merely provide ports as argued by the Appellant.

The Appellant further argues that Rompaey does not teach "generating said FSM representation comprises applying a design language having at least one graphical symbol and adapted to form a finite state machine representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol".

Rompaey recites Fig. 11 on p. 31, text at p. 8, col. 11, lines 5-6:

> Figure 11 is a schematic representation of the pager application as described with the present invention.

Figure 11 in combination with recitation at col. 9 lines 14-27 clearly teaches graphical symbol and adapted to form a finite state machine representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion inside the symbol defining the behavior of the graphical symbol. For example, graphical symbol 97 (bottom right corner) represents Phase Correction design with design language textual portion 110 inside it to define its behavior.

The Appellant also argues that Cadence does not teach " creating a virtual test bench having a graphical representation of a human/machine interface for selecting signals or variables associated with the FSM and to be coupled to a graphical representation of a user interface for each interactive test-bench". The Examiner respectfully disagrees.

Cadence recites on p. 2, paragraphs 1-3:

The Interactive Simulation Library makes constructing VIs of any kind simple and straightforward. ISL control windows can display scroll bars, push-buttons, pull-down menus and more for changing gains, frequencies, noise-levels and other design parameters. ISL display windows show real-time oscillograms, bar graphs, eye diagrams, and waterfall plots indicating the effects the controls have on the system.

You specify features and displays for an interactive control window by choosing ISL primitives (basic function blocks) from the library and wiring them together on the workstation screen (hierarchy is allowed). The resulting block diagram is a complete blueprint for the window; no written programming is required. These capabilities enable you to define whatever kind of signal post-processing is required to display the information you want.

Once the ISL block diagram is specified, you incorporate it into the signal flow block diagram for the system under test. When you run a simulation, the interactive control and display windows you created pop up automatically and begin monitoring the systems performance.

In the above paragraphs Cadence teaches a test-bench, a tool to test a design, that a user can interactively select signals on a design, which can be a FSM when combined with Rompaey's teachings, to be post-processed and displayed after running a simulation.

It would have been obvious to one of ordinary skill in the art to combine the teachings of Rompaey, Cadence, and Van Huben. Cadence's and Van Huben's teachings would have provided access of the stored embedded system for reuse to a selected group or people, and interactive control of the simulation and visual examination of outcome rather than in "batch-mode" and dramatically accelerated the integration, test and evaluation phase of product development (Cadence, p. 3, paragraph 1).

Claim 68, therefore, is rejected.

12. As per claims 71-72, the Appellant argues that they are allowed since it depends on allowed claim 68, see p. 35. Since claim 68 remains rejected, claims 71-72 remain rejected.

13. The Appellant's arguments, regarding claim 74, see pages 36-39 of Appeal Brief, have been fully considered but they are not persuasive. The Appellant argues that Rompaey does not

teach "... the virtual prototypes having a processor simulator, a finite state machine (FSM) representation of hardware peripherals;

wherein generating said FSM representation comprises applying a design language having at least one graphical symbol and adapted to form a finite state machine representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol".

The Examiner respectfully disagrees. At col. 9 lines 14-27, Rompaey recites:

> Said simulation can be a multi-platform simulation being executed on a plurality of computers.
> Said simulation can be a hybrid simulation comprising substantially simultaneous hardware implementations and computer simulations.
> Said implementation can be heterogeneous implementation comprising hardware subsystems and software subsystems, said software subsystems being executed on one on more of said hardware subsystems.
> Said hardware subsystems can comprise any one or more of processor cores, off-the-shelf components, custom components, ASICs, processors, or boards.
> Said software subsystems can comprise machine instructions for said hardware subsystems

and Fig. 11 on p. 31, text at col. 11, lines 5-6:

> Figure 11 is a schematic representation of the pager application as described with the present invention.

Rompaey, as disclosed on col. 9 lines 14-27, teaches hardware subsystems comprising processor cores for simulation. A processor core is made up of the control unit and arithmetic logic unit. The control unit is further divided into 2 sub units, one of which is a finite state machine (FSM), which is used to control sequences of all actions performed and elements for producing proper control signals for difference elements. Rompaey also teaches machine instructions for said hardware subsystems. This teaching in combination

with the teaching of processor cores for hardware subsystems read onto the limitation of an instruction set accurate simulator of a target processor core.

Figure 11 in combination with recitation at col. 9 lines 14-27 clearly teaches graphical symbol and adapted to form a finite state machine representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion inside the symbol defining the behavior of the graphical symbol. For example, graphical symbol 97 (bottom right corner) represents Phase Correction design with design language textual portion 110 inside it to define its behavior.

The Appellant also argues that Cadence does not teach "the virtual test bench is created using a test bench builder for generating a graphical representation of at least one interactive test bench and for selecting signals or variables associated with the FSM and to be coupled to a graphical representation of a user interface for each interactive test-bench". The Examiner respectfully disagrees. Cadence recites on p. 2, paragraphs 1-3:

> The Interactive Simulation Library makes constructing VIs of any kind simple and straightforward. ISL control windows can display scroll bars, push-buttons, pull-down menus and more for changing gains, frequencies, noise-levels and other design parameters. ISL display windows show real-time oscillograms, bar graphs, eye diagrams, and waterfall plots indicating the effects the controls have on the system.
>
> You specify features and displays for an interactive control window by choosing ISL primitives (basic function blocks) from the library and wiring them together on the workstation screen (hierarchy is allowed). The resulting block diagram is a complete blueprint for the window; no written programming is required. These capabilities enable you to define whatever kind of signal post-processing is required to display the information you want.
>
> Once the ISL block diagram is specified, you incorporate it into the signal flow block diagram for the system under test. When you run a simulation, the interactive control and display windows you created pop up automatically and begin monitoring the systems performance.

In the above paragraphs Cadence teaches a test-bench, a tool to test a design, that a user can interactively select signals on a design, which can be a FSM when combined with Rompaey's teachings, to be post-processed and displayed after running a simulation.

It would have been obvious to one of ordinary skill in the art to combine the teachings of Rompaey, Cadence, and Van Huben. Cadence's and Rompaey's teachings would have made a virtual prototype of embedded system accessible for evaluation and provided interactive control of the simulation and visual examination of outcome rather than in "batch-mode" and dramatically accelerated the integration, test and evaluation phase of product development. (Cadence, p. 3, paragraph 1).

Claim 74, therefore, is rejected.

14. As per claim 75, the Appellant argues that they are allowed since it depends on allowed claim 74, see p. 39. Since claim 74 remains rejected, claim 75 remains rejected.

15. As per claims 76-78, the Appellant argues that they are allowed since it depends on allowed claim 74, see p. 39. Since claim 74 remains rejected, claims 76-78 remain rejected.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

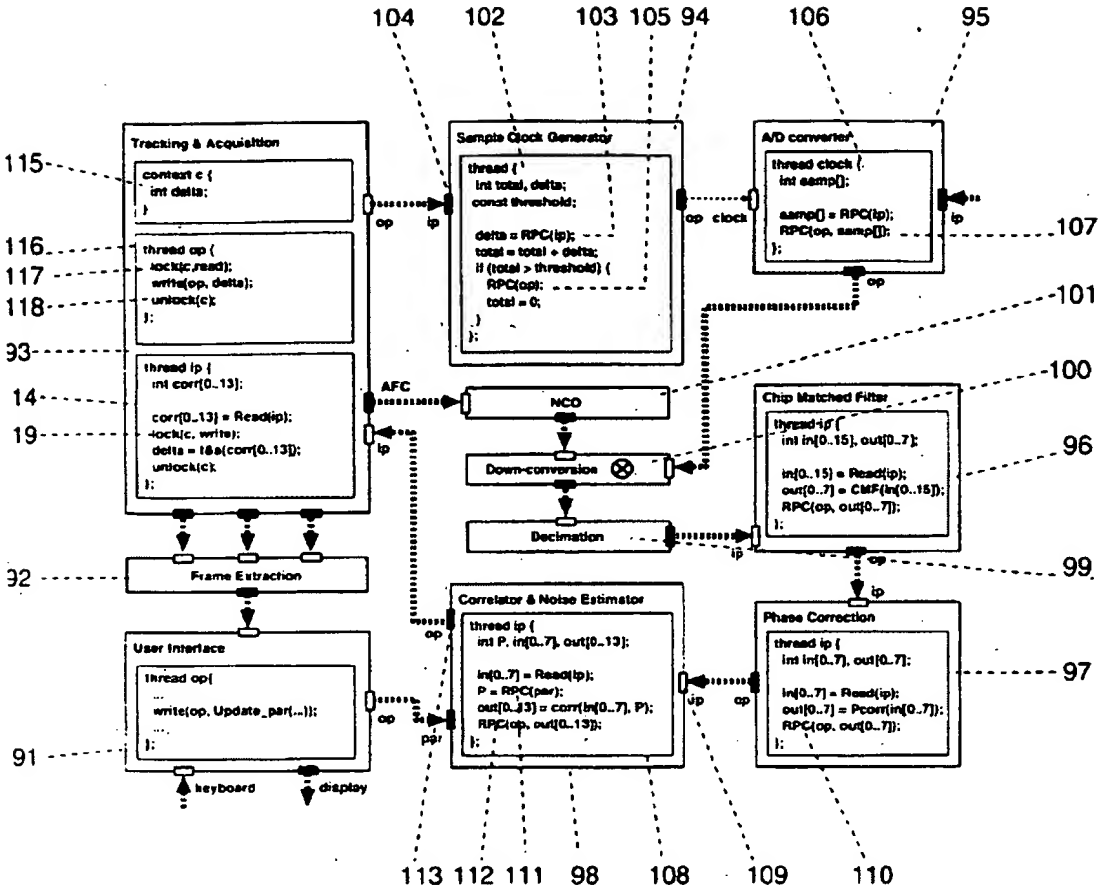

FIGURE 11.

SDOCID: <EP 0772140A1 I >

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Cuong Luu
Patent Examiner

Conferees:
Eddie Lee
TQAS/Appeals Specialist, TC 2100

EDDIE C. LEE
SUPERVISORY PATENT EXAMINER

Kamini Shah
SPE AU 2128

Jeanette Harms
Registration No. 35,537
Attorney for Applicant(s)